Unbreakable DevOps on Red Hat OpenShift

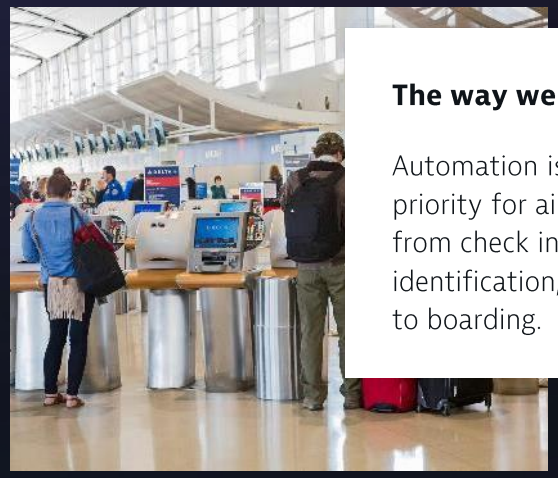
Peter Hack
Technology Strategist

Florian Bacher
Technology Strategist

Intro – today's world

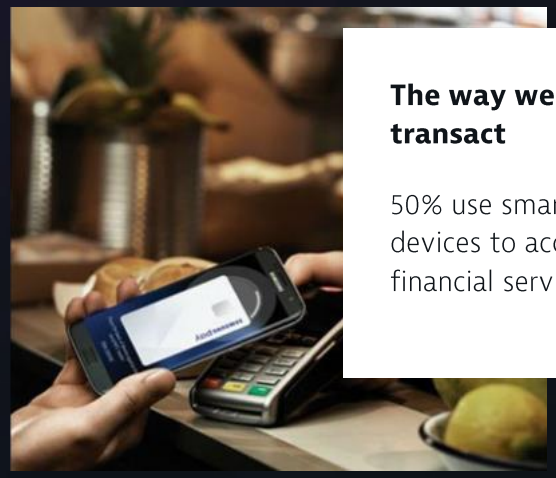
What is the Challenge to Unbreakable DevOps?

Digital transformation is software defined



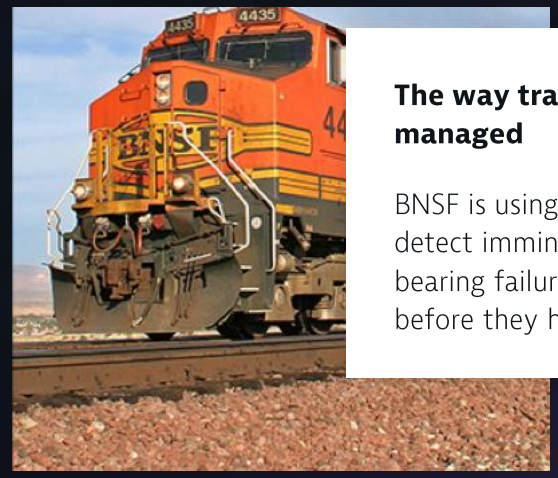
The way we fly

Automation is a key priority for airlines from check in, to identification, to boarding.



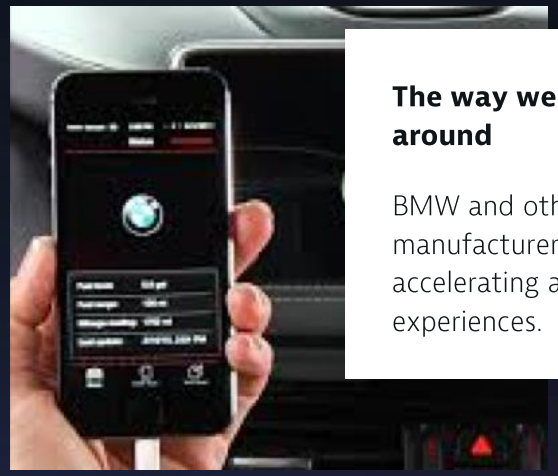
The way we transact

50% use smart devices to access financial services.



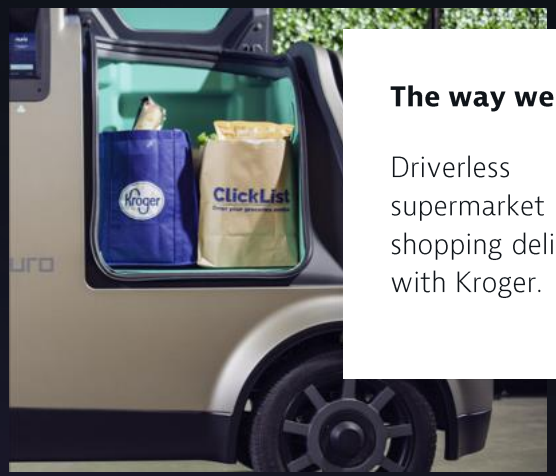
The way trains are managed

BNSF is using IoT to detect imminent bearing failures before they happen.



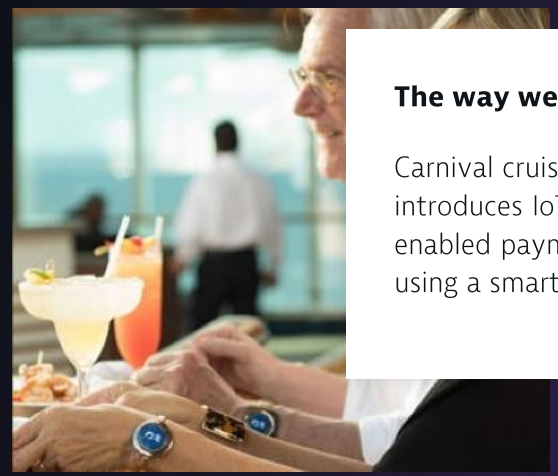
The way we get around

BMW and other car manufacturers accelerating application experiences.



The way we shop

Driverless supermarket shopping delivery with Kroger.



The way we play

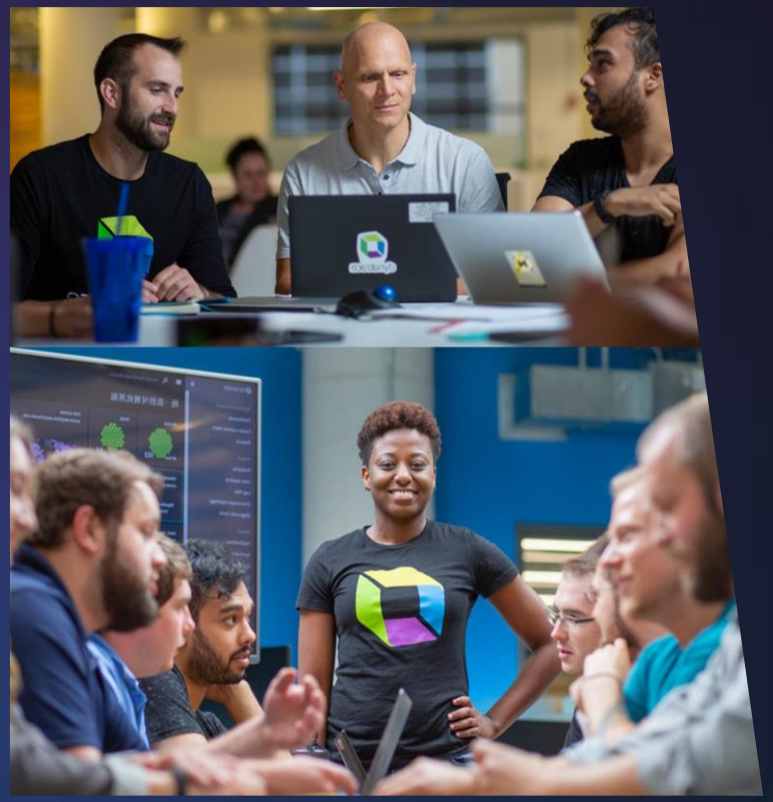
Carnival cruises introduces IoT enabled payment using a smartwatch.

And the software driving the business needs to work P E R F E C T L Y

Our own journey

We have done this and so can you

Who is Dynatrace?



OpenShift Primed Partner

Technology partner

Docker Certified

AWS Advanced Technology Partner

Azure Partner

9x

Gartner Magic Quadrant leader 9 years in a row

#1

Market Share (Gartner, IDC)

750%

YoY Growth of the new all-in-one platform

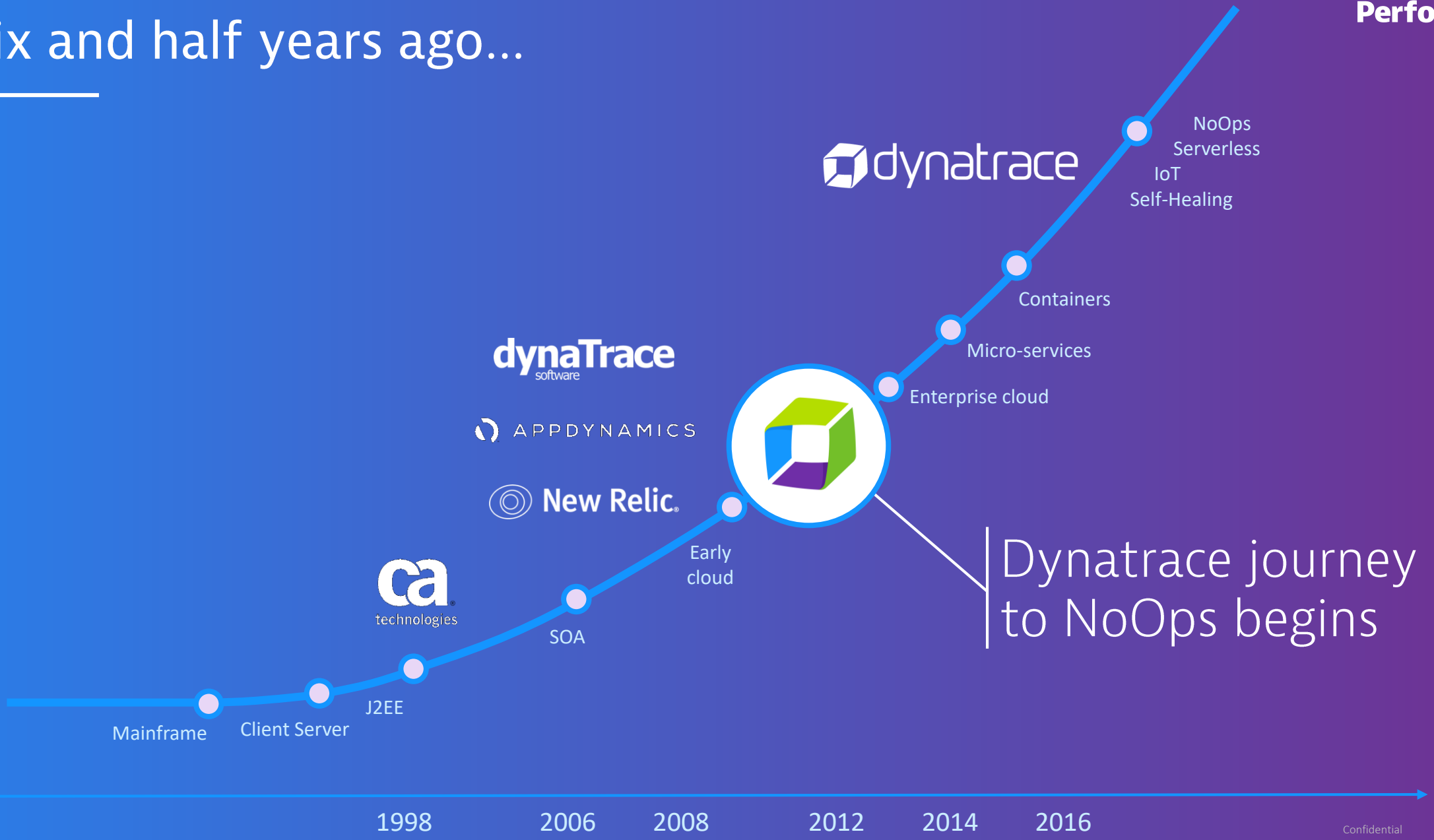
26

Major releases per year

72 / 100

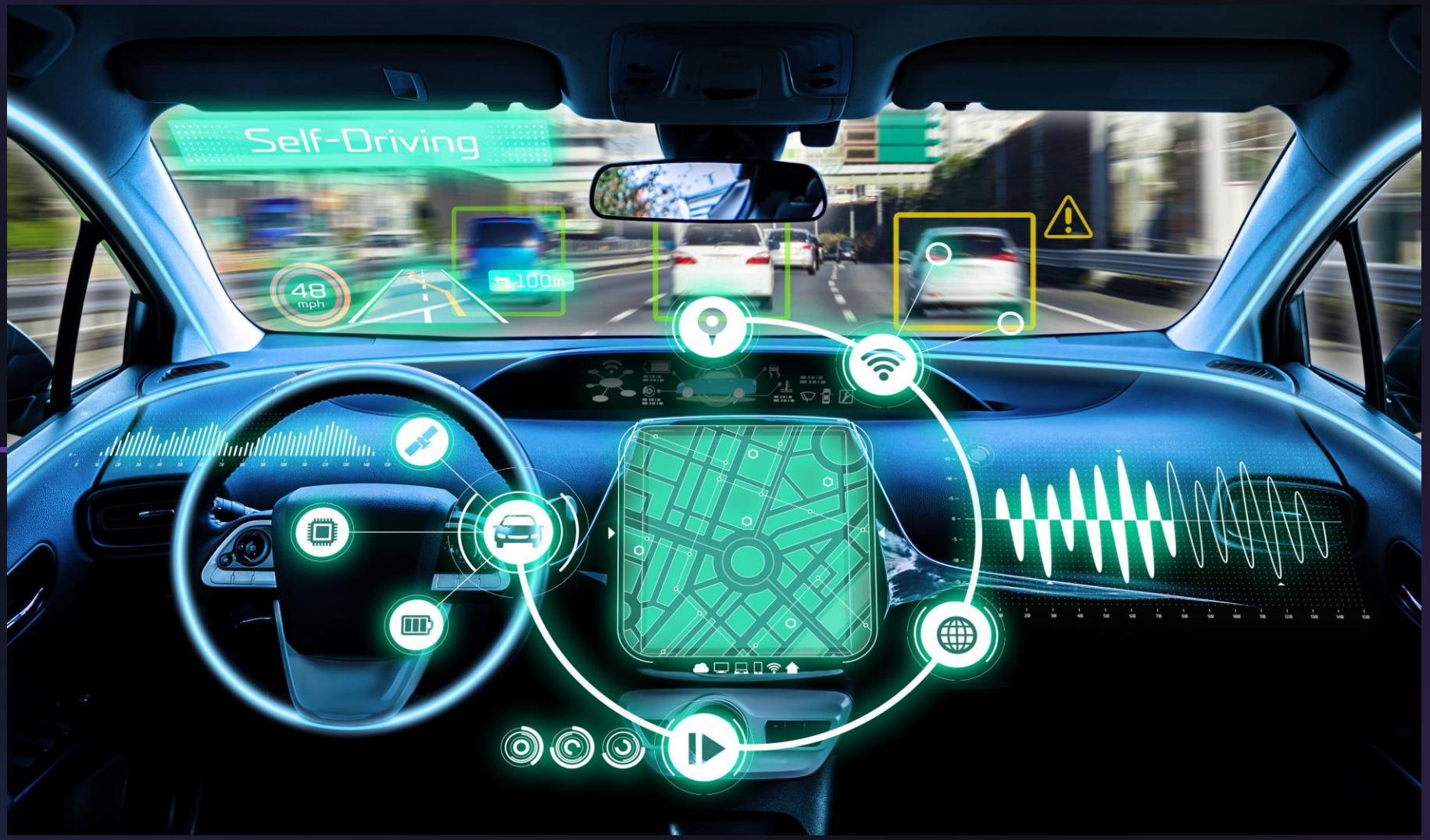
Customers in Fortune 100

Six and half years ago…



Dynatrace journey to NoOps begins

Vision of “self-driving” DevOps …



Achieving, seeing, sharing success!

releases per year

production bugs reported by customers

EC2 instances

daily deployments

Your App / Container

Pivotal Cloud Foundry

OPENSHIFT

kubernetes

aws

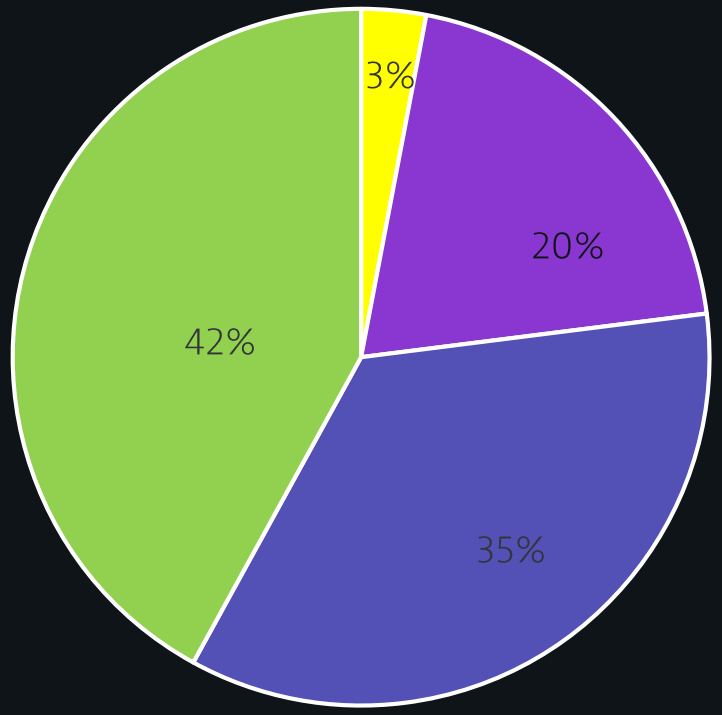
Google Cloud Platform

Azure

SAP S/4 HANA

Cloud Native

Collecting more evidence: https://dynatrace.ai/acsurvey



- Small (11-100 employees)
- Medium (101-1000 employees)
- Large (1001-5000 employees)
- Extra large (over 5000 employees)

Commit Cycle Time: From Dev to Pro

Verdict: The Majority is not delivering high quality faster

	Code to Production (Commit Cycle Time)	Business **Impacting** Deployments	**Per** Production **Deployment**	MTTR (Mean Time to Repair)
95th Percentile	2 days	1 out of 10	0 hotfixes	~4 hours
Median	12.5 days	3 out of 10	3 hotfixes	4.8 days

Evaluate for yourself: https://dynatrace.ai/acsurvey

How does an organization from Median to 95th percentile!

Faster

97%
deployment lead time

More Frequently

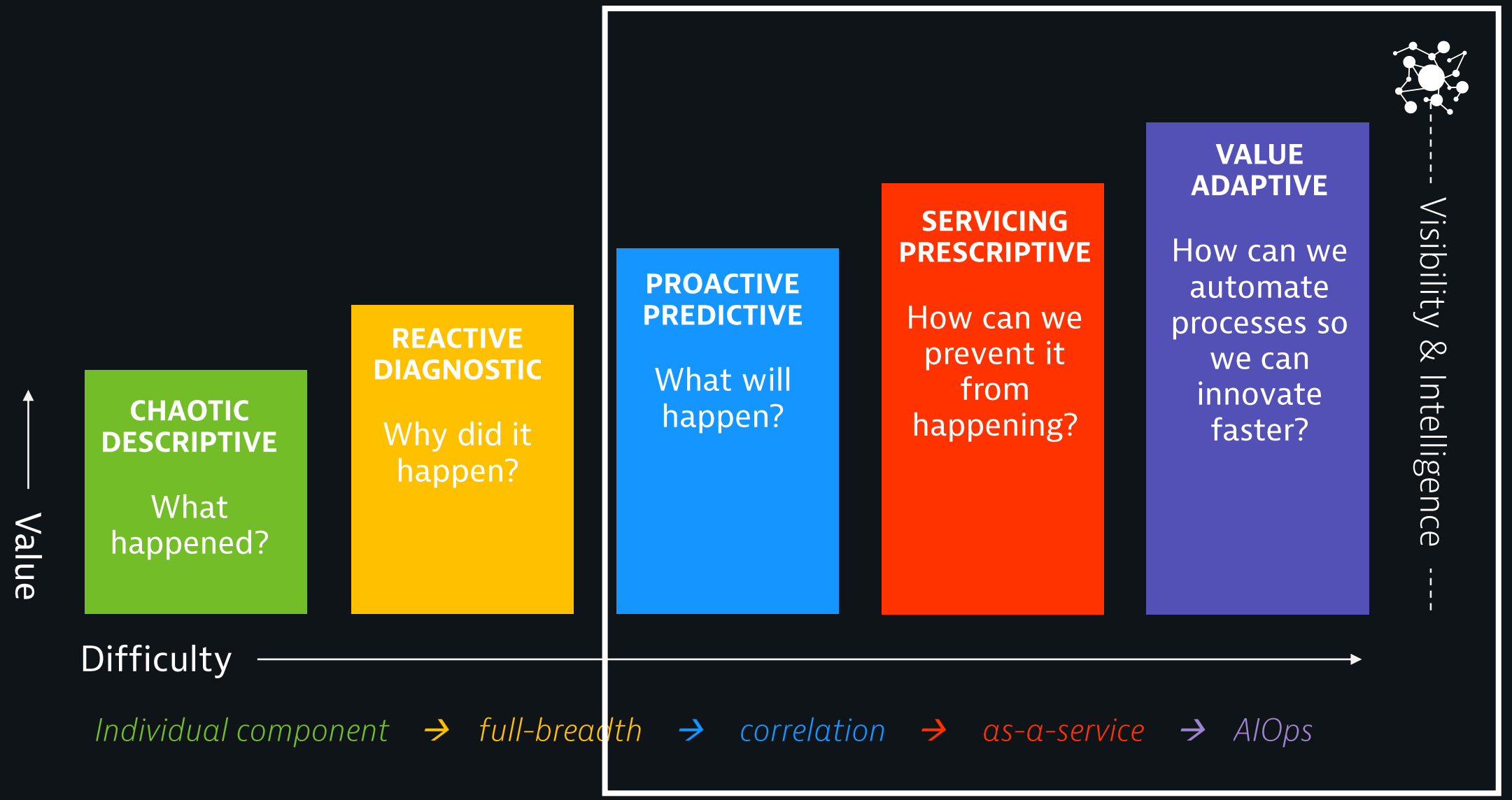
12 to 26
releases per year

Better

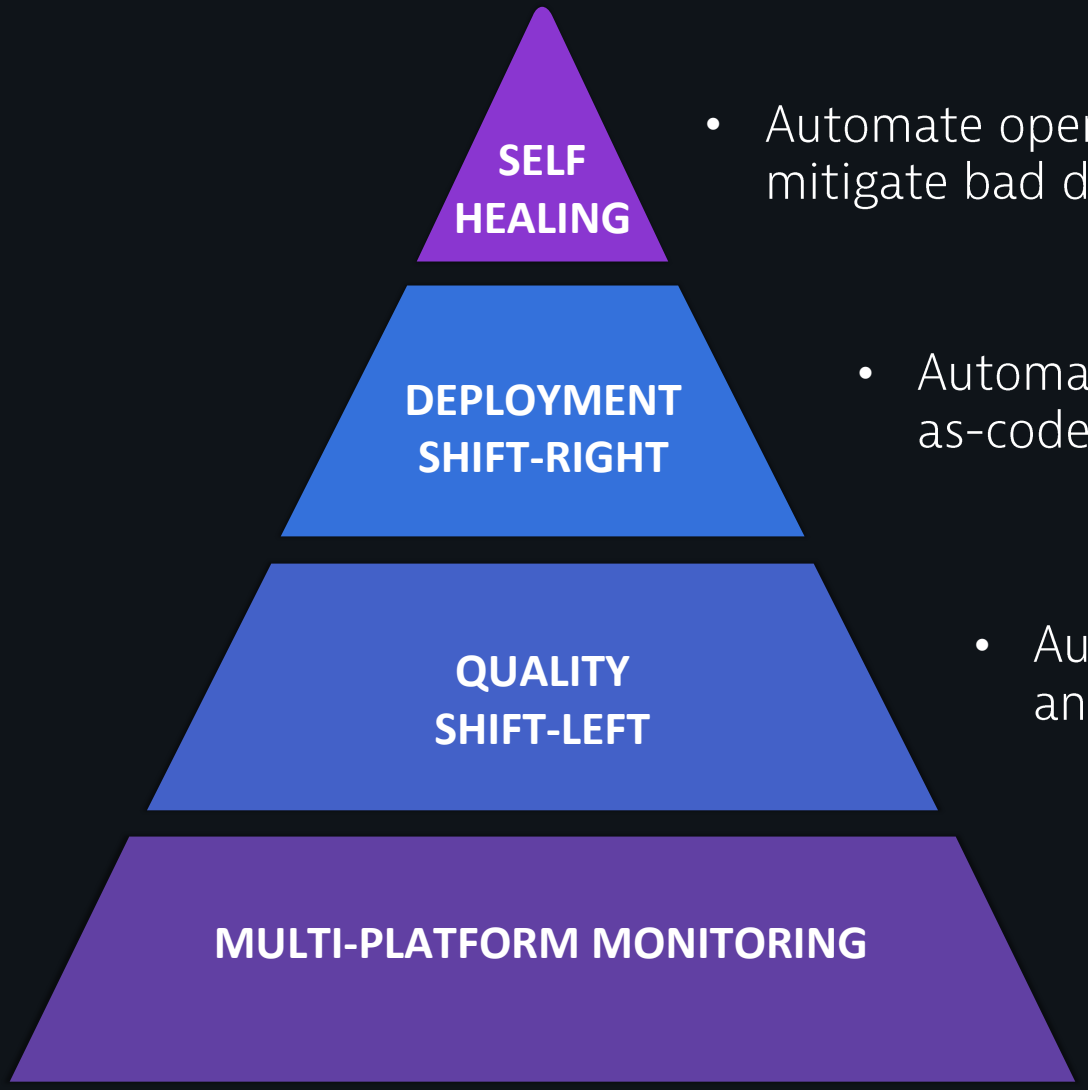
75%
production incidents

- Automated Quality (Shift-Left)
  - **Goal**: Stop Bad Code Changes Early & Automated
  - Automated Testing & Quality Gates
- Automated Multi-Stage Deployments (Shift-Right)
  - **Goal**: Increase deployments into **stable** environments
  - Dark, Shadow, Blue/Green Deployments with Auto-Validation
- Automated Operations (Self-Healing)
  - **Goal**: NoOps & Zero-Impact on End Users
  - Automated Remediation & (On-Demand) Scaling

The DevOps journey has a world-class monitoring roadmap



The DevOps journey is built on a solid foundation



- Automate operations (self-healing) – auto-mitigate bad deployments in production
- Automate deployment (shift-right) – push "monitoring-as-code" for auto-validation and auto-alerting
- Automate quality (shift-left) – automate the pipeline and stop bad code changes before they reach prod
- Automated monitoring – monitoring as feature of the end-to-end pipeline

delivering better software faster

“Any journey begins with the first step”

The first step

Requirements

Checkout

Build

Packaging

Delivery

Feedback

Checkout

Build

Packaging

Delivery

Is the order correct?

Yes

no

Requirements

Checkout

correct Ingredients?

Yes

no

Build

Cooked Correctly?

Yes

no

Packaging

All Items Prepared?

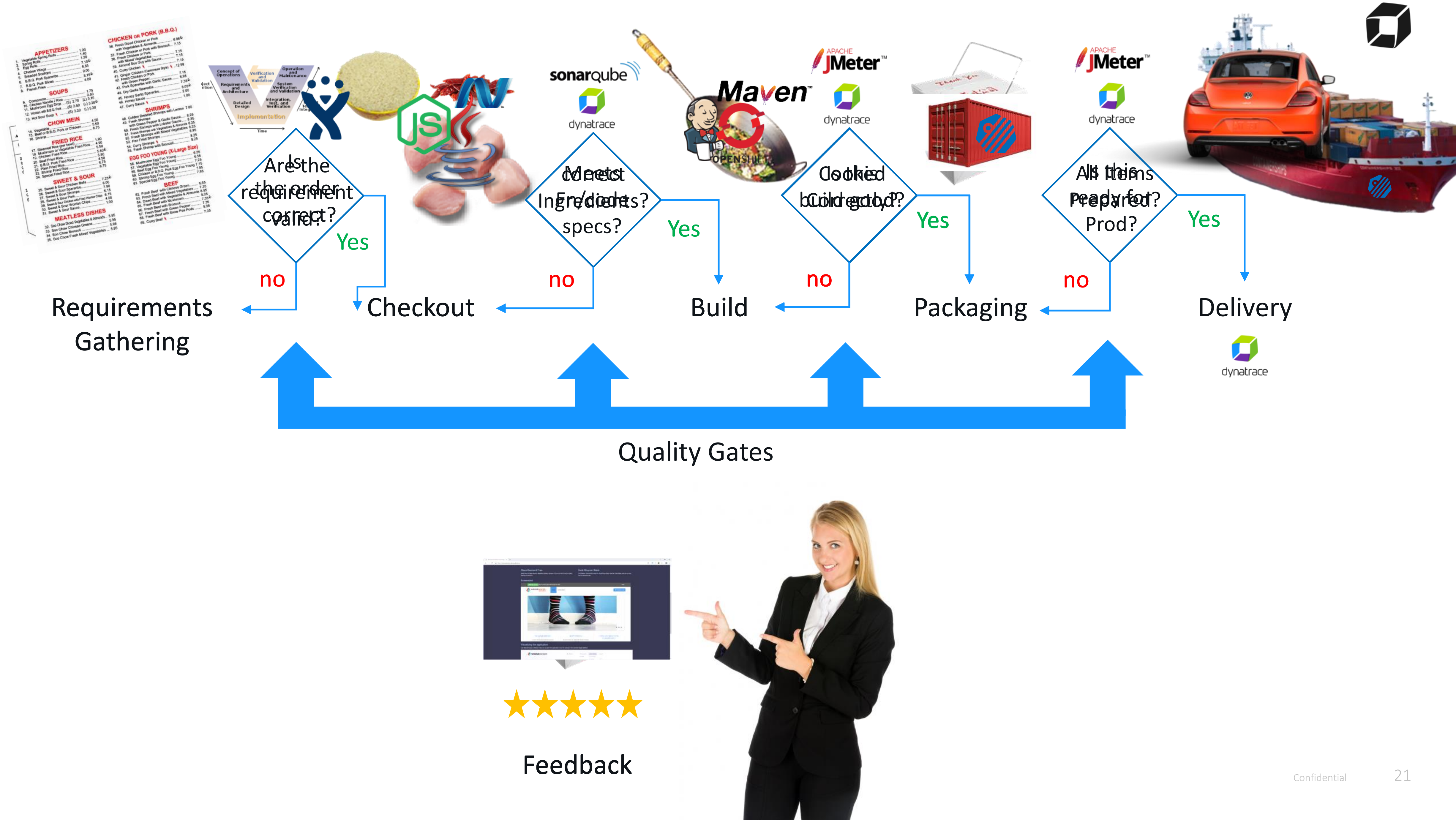
Yes

no

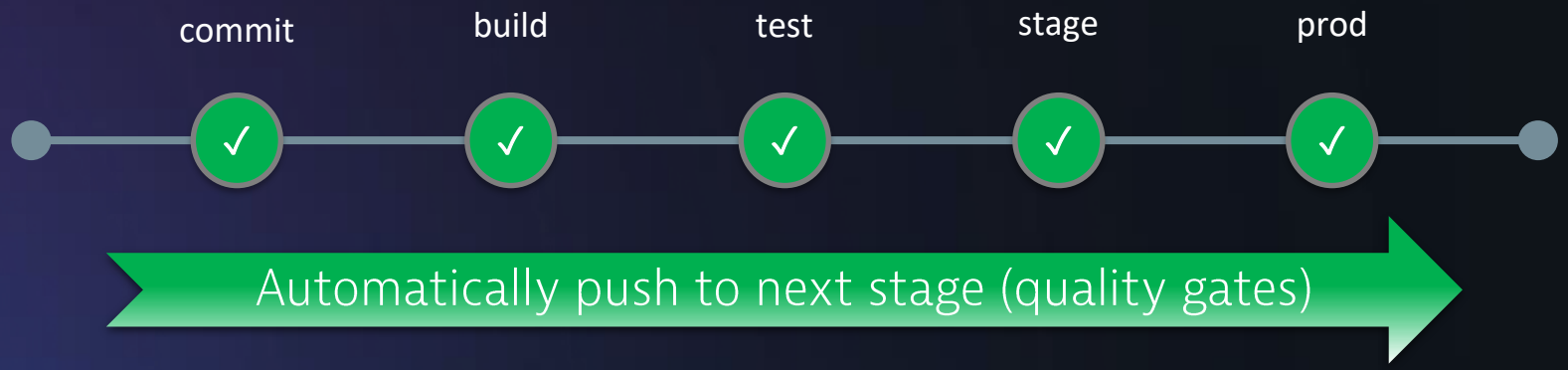
Delivery

★★★★★

Feedback


sonarqube
dynatrace
Maven
OPENSHIFT
APACHE JMeter
dynatrace
APACHE JMeter
dynatrace
Yes
Yes
Yes
Yes
no
no
no
no
specs?
Prod?
Requirements Gathering
Checkout
Build
Packaging
Delivery
dynatrace
Quality Gates
Feedback

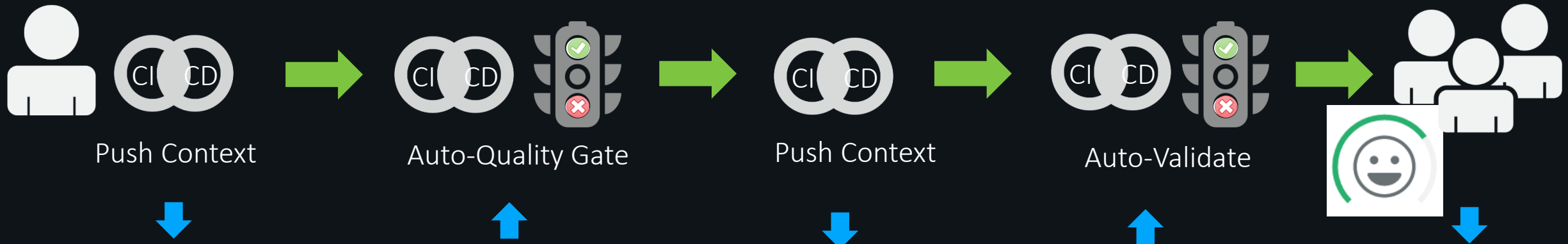
What we want



What we (sometimes) have

Unbreakable delivery pipeline in action

1 Staging → 2 Approve Staging → 3 Production → 4 Approve Production → 5

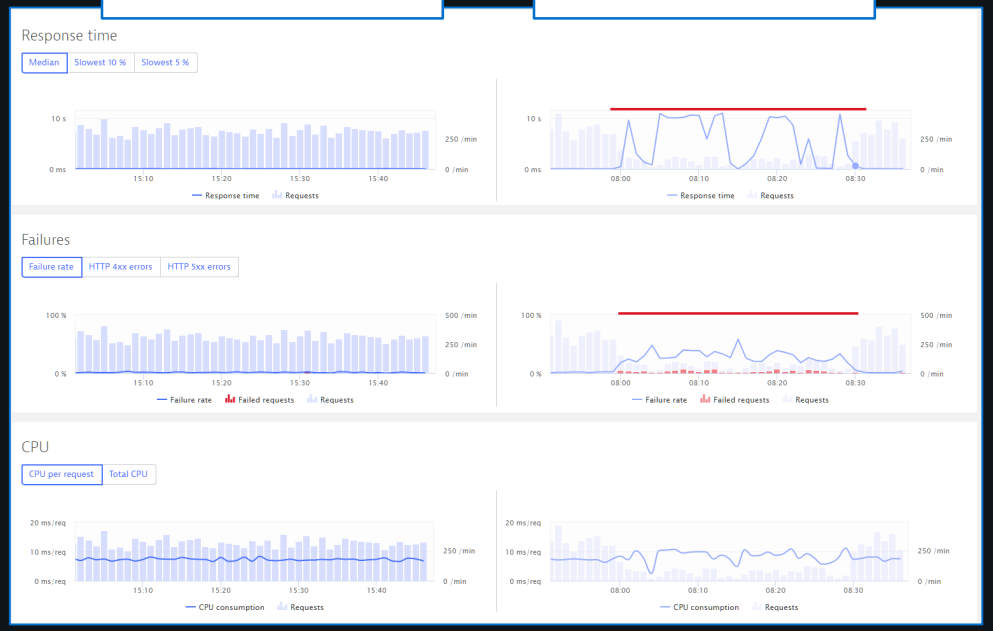


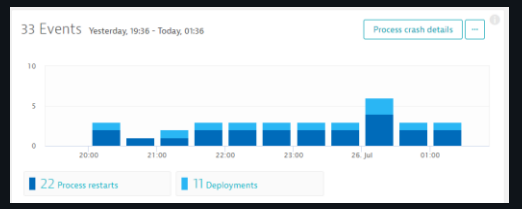
Push Context

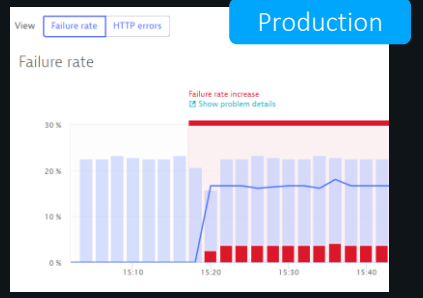
Auto-Quality Gate

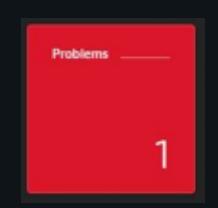
Push Context

Auto-Validate

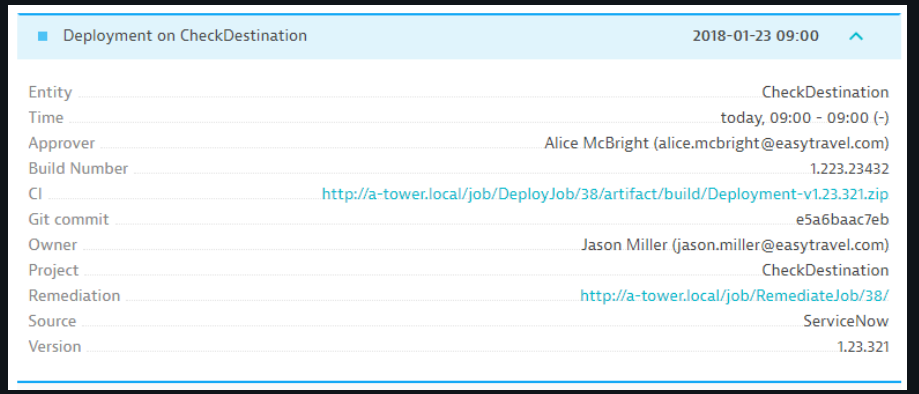


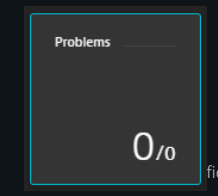


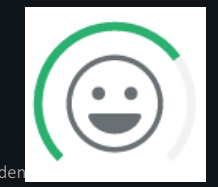




Auto-Remediate!







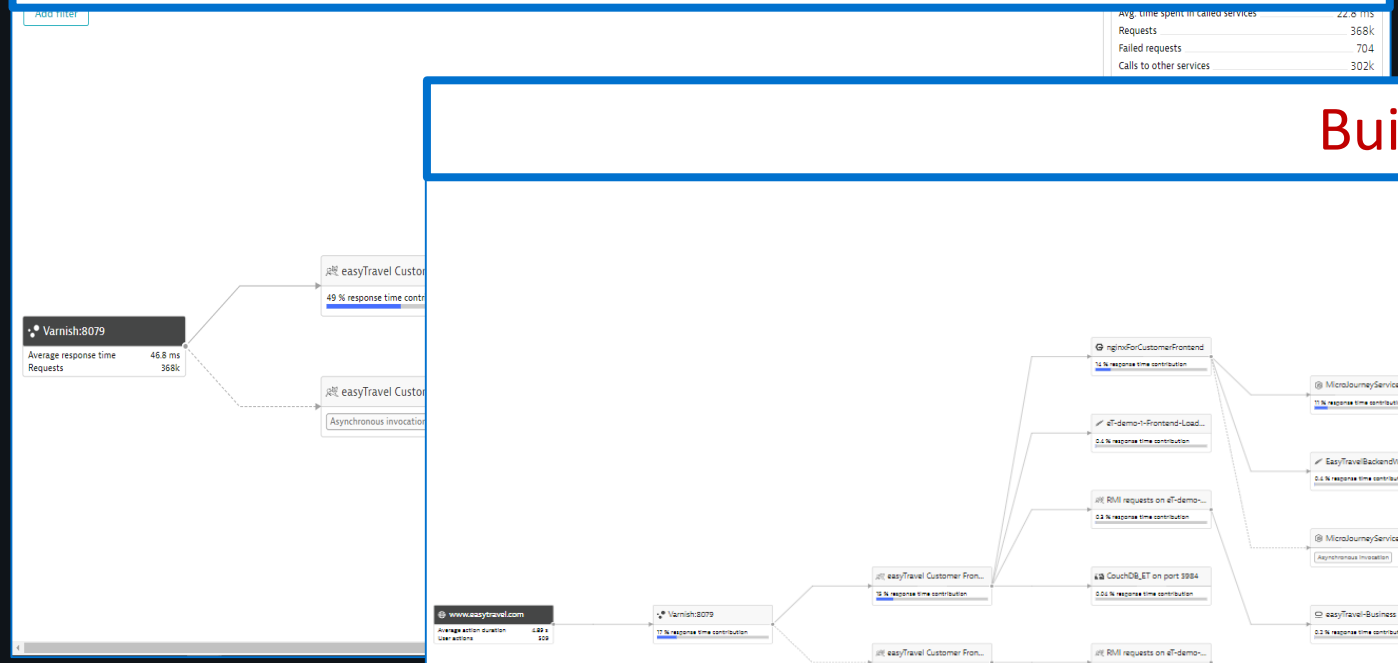
Automated Deployment Validation

https://github.com/keptn/pitometer

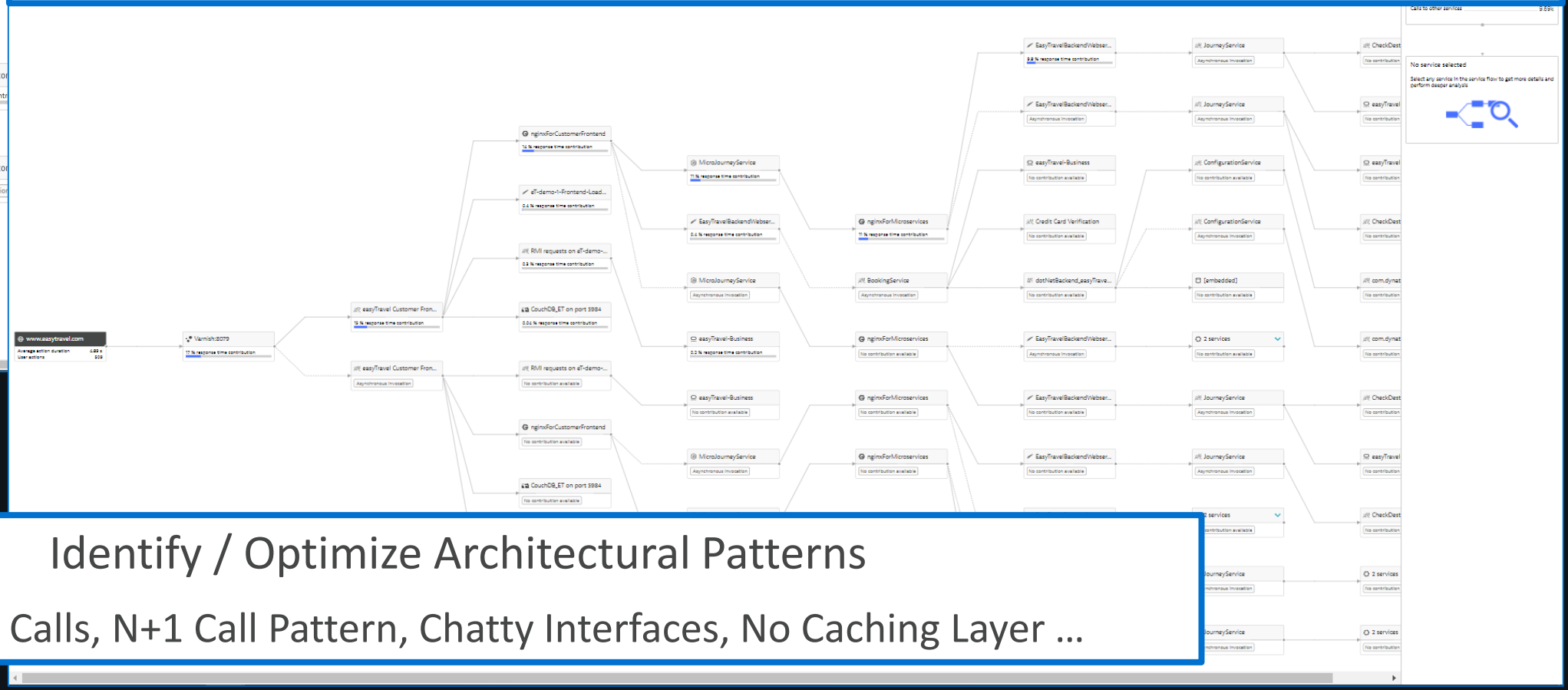
https://github.com/keptn,
www.keptn.sh

Automate Architectural Checks into CI/CD/CO!

Build #17



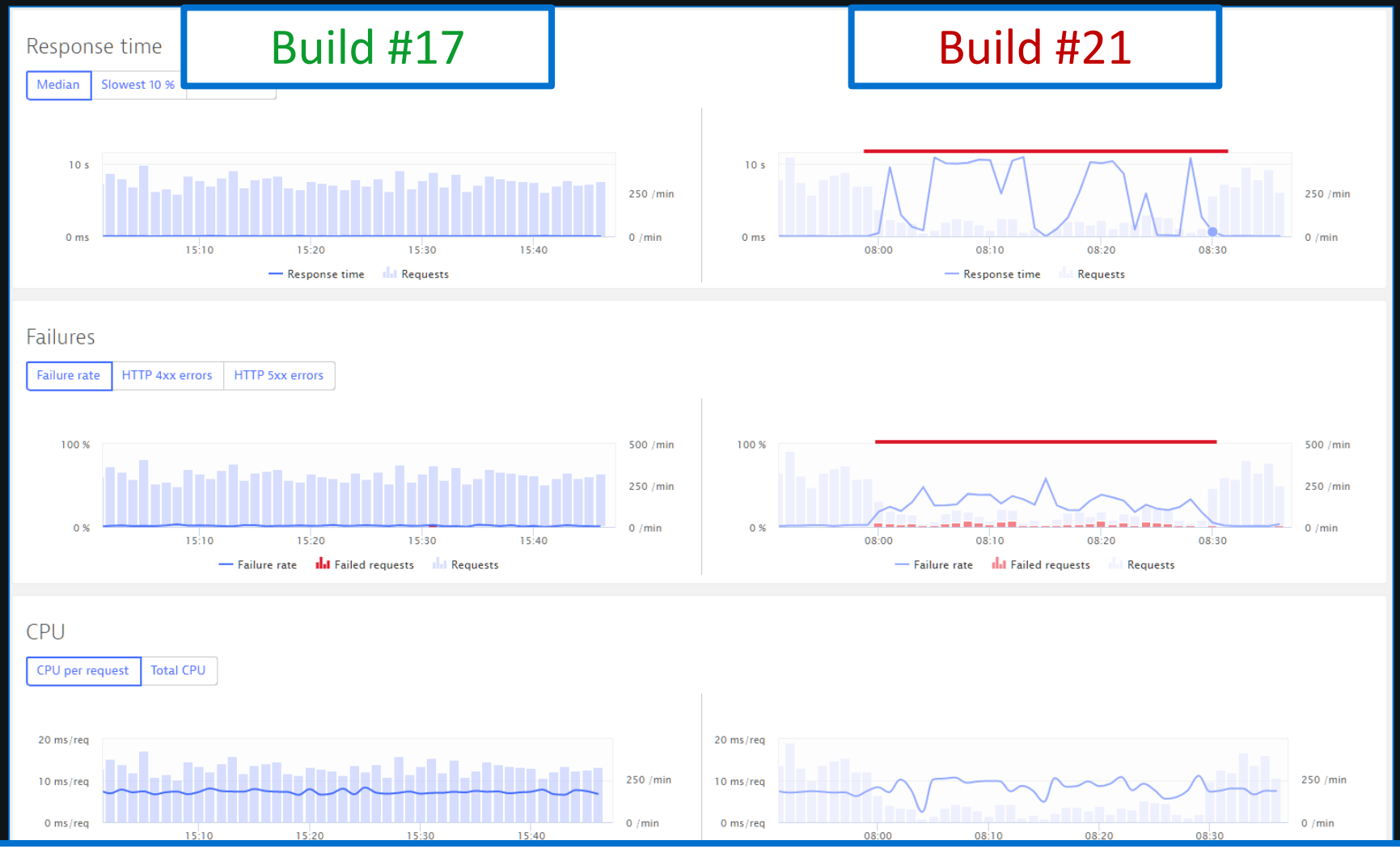
Build #21



Identify / Optimize Architectural Patterns

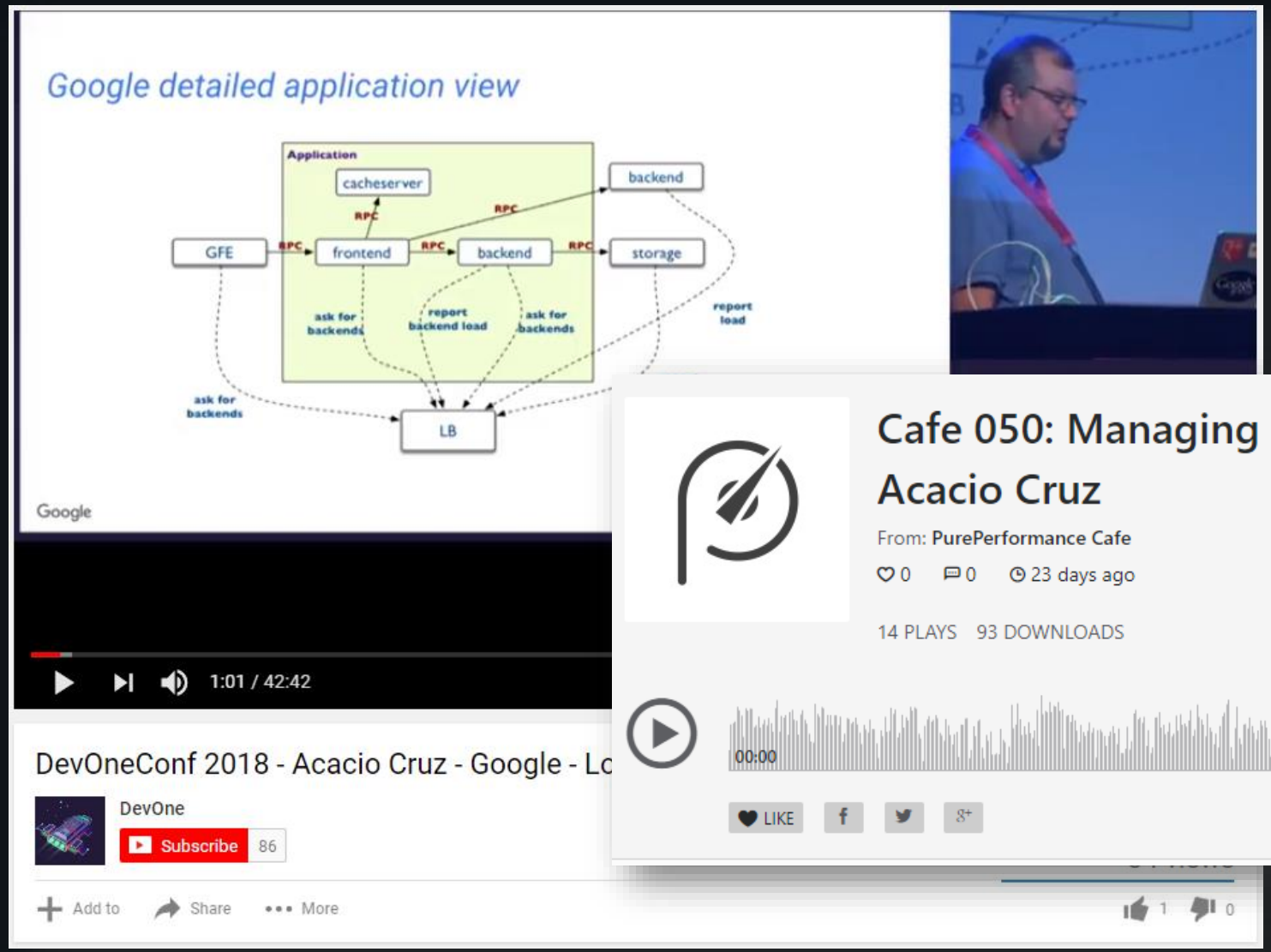
Recursive Calls, N+1 Call Pattern, Chatty Interfaces, No Caching Layer …

Automate Performance Checks into CI/CD/CO!

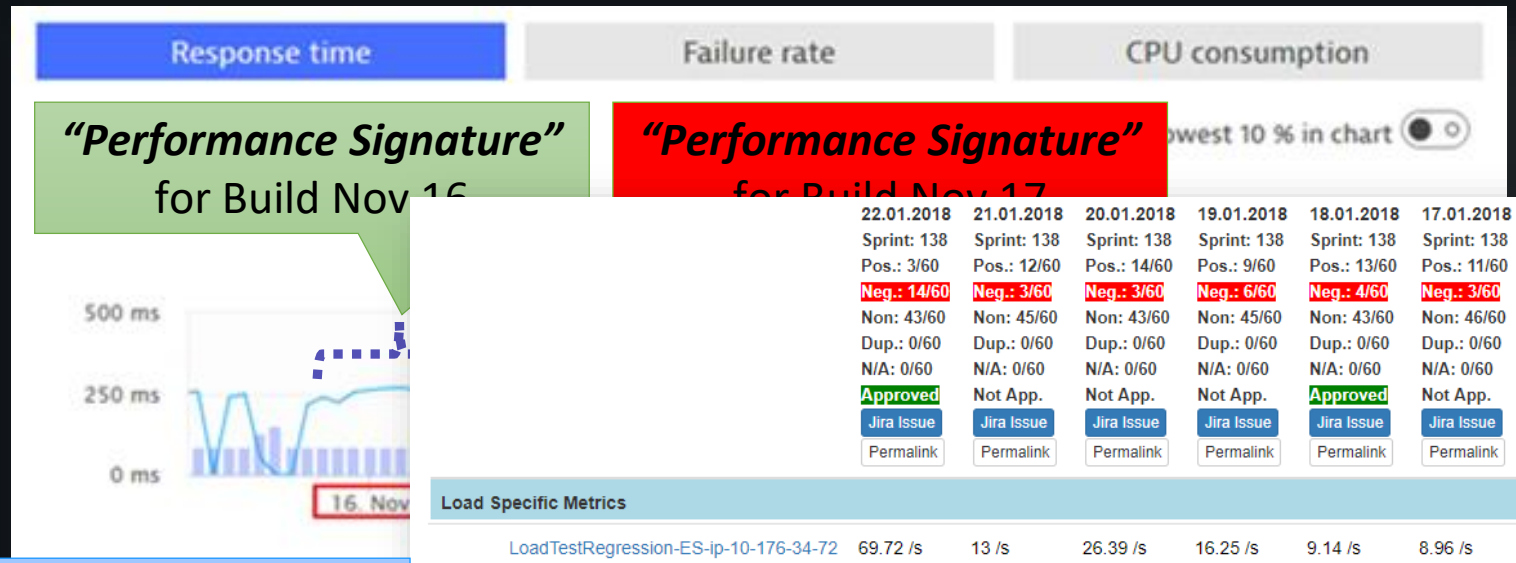


How is Performance & Resource Consumption per Service Endpoint?

From Google: “Everything as Code” e.g: Enforce Architectural Rules



From Dynatrace: „Performance Signature as Code“ evaluated through Jenkins



“Performance Signature” for every Build

“Multiple Metrics” compared to prev Timeframe

	22.01.2018	21.01.2018	20.01.2018	19.01.2018	18.01.2018	17.01.2018	16.01.2018	15.01.2018	14.01.2018	13.01.2018	12.01.2018	11.01.2018	10.01.2018	09.01.2018
	Sprint: 138 Pos.: 3/60 Neg.: 14/60 Non: 43/60 Dup.: 0/60 N/A: 0/60 Approved Jira Issue Permalink	Sprint: 138 Pos.: 12/60 Neg.: 3/60 Non: 45/60 Dup.: 0/60 N/A: 0/60 Not App. Jira Issue Permalink	Sprint: 138 Pos.: 14/60 Neg.: 3/60 Non: 43/60 Dup.: 0/60 N/A: 0/60 Not App. Jira Issue Permalink	Sprint: 138 Pos.: 9/60 Neg.: 6/60 Non: 45/60 Dup.: 0/60 N/A: 0/60 Not App. Jira Issue Permalink	Sprint: 138 Pos.: 13/60 Neg.: 4/60 Non: 43/60 Dup.: 0/60 N/A: 0/60 Approved Jira Issue Permalink	Sprint: 138 Pos.: 11/60 Neg.: 3/60 Non: 46/60 Dup.: 0/60 N/A: 0/60 Not App. Jira Issue Permalink	Sprint: 138 Pos.: 13/60 Neg.: 2/60 Non: 45/60 Dup.: 0/60 N/A: 0/60 Not App. Jira Issue Permalink	Sprint: 138 Pos.: 8/60 Neg.: 8/60 Non: 44/60 Dup.: 0/60 N/A: 0/60 Not App. Jira Issue Permalink	Sprint: 138 Pos.: 12/60 Neg.: 6/60 Non: 42/60 Dup.: 0/60 N/A: 0/60 Not App. Jira Issue Permalink	Sprint: 138 Pos.: 15/60 Neg.: 5/60 Non: 40/60 Dup.: 0/60 N/A: 0/60 Not App. Jira Issue Permalink	Sprint: 138 Pos.: 15/60 Neg.: 6/60 Non: 39/60 Dup.: 0/60 N/A: 0/60 Not App. Jira Issue Permalink	Sprint: 138 Pos.: 15/60 Neg.: 7/60 Non: 38/60 Dup.: 0/60 N/A: 0/60 Approved Jira Issue Permalink	Sprint: 138 Pos.: 13/60 Neg.: 7/60 Non: 40/60 Dup.: 0/60 N/A: 0/60 Approved Jira Issue Permalink	Sprint: 137 Pos.: 17/60 Neg.: 5/60 Non: 38/60 Dup.: 0/60 N/A: 0/60 Approved Jira Issue Permalink
Load Specific Metrics														
LoadTestRegression-ES-ip-10-176-34-72 /media/datastore: Timeseries: IOPS Reads(avg) Thresholds: Rel: 10.0; Max: Not Set; Min: Not Set; Amp: Not Set Average over the last 14 Days: 24.34 /s Permalink	69.72 /s A: (--; --), L: (56.72 /s; 436.28 %), 14d: 186.49 % Link Permalink	13 /s A: (-56.72 /s; -81.35 %), L: (-13.39 /s; -50.73 %), 14d: -46.58 % Link Permalink	26.39 /s A: (-43.33 /s; -62.15 %), L: (10.14 /s; 62.39 %), 14d: 8.43 % Link Permalink	16.25 /s A: (-53.47 /s; -76.69 %), L: (7.11 /s; 77.82 %), 14d: -33.23 % Link Permalink	9.14 /s A: (-60.58 /s; -86.89 %), L: (0.18 /s; 2.02 %), 14d: -62.45 % Link Permalink	8.96 /s A: (-60.76 /s; -87.15 %), L: (-3.37 /s; -27.35 %), 14d: -63.2 % Link Permalink	12.33 /s A: (-57.39 /s; -82.32 %), L: (-61.25 /s; -83.24 %), 14d: -49.34 % Link Permalink	73.57 /s A: (3.86 /s; 5.53 %), L: (53.14 /s; 260.06 %), 14d: 202.33 % Link Permalink	20.43 /s A: (-49.29 /s; -70.69 %), L: (-6.04 /s; -22.82 %), 14d: -16.03 % Link Permalink	26.48 /s A: (-43.24 /s; -62.03 %), L: (-2.45 /s; -8.47 %), 14d: 8.79 % Link Permalink	28.92 /s A: (-40.8 /s; -58.51 %), L: (18.57 /s; 179.37 %), 14d: 18.85 % Link Permalink	10.35 /s A: (-59.37 /s; -85.15 %), L: (-1.2 /s; -10.35 %), 14d: -57.46 % Link Permalink	11.55 /s A: (-58.17 /s; -83.44 %), L: (-2.07 /s; -15.18 %), 14d: -52.54 % Link Permalink	13.62 /s A: (-56.1 /s; -80.47 %), L: (0 /s; 0 %), 14d: -44.05 % Link Permalink
LoadTestRegression-ES-ip-10-176-34-72 /media/datastore: Timeseries: Throughput Writes(avg) Min: Not Set; Amp: Not Set	6331.79 KB/s A: (--; --), L: (3503.02 KB/s; 123.84 %),	2828.76 KB/s A: (-3503.02 KB/s; -55.32 %),	3351.54 KB/s A: (-2980.25 KB/s; -47.07 %),	3092.48 KB/s A: (-3239.31 KB/s; -51.16 %),	2894.31 KB/s A: (-3437.48 KB/s; -54.29 %),	3001.3 KB/s A: (-3330.49 KB/s; -52.6 %),	3230.92 KB/s A: (-3100.87 KB/s; -48.97 %),	6515.71 KB/s A: (183.93 KB/s; 2.9 %), L: (3168.31	3347.41 KB/s A: (-2984.38 KB/s; -47.13 %),	3559.69 KB/s A: (-2772.1 KB/s; -43.78 %), L: (15.53	3544.16 KB/s A: (-2787.63 KB/s; -44.03 %),	2946.86 KB/s A: (-3384.92 KB/s; -53.46 %),	3095.73 KB/s A: (-3236.06 KB/s; -51.11 %),	3109.93 KB/s A: (-3221.86 KB/s; -50.88 %),

Simple Regression Detection per Metric

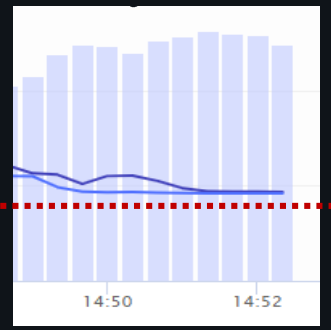
Pitometer (part of @keptnProject): Metrics-based grading of a Deployment!

Pitometer Specfile

Source

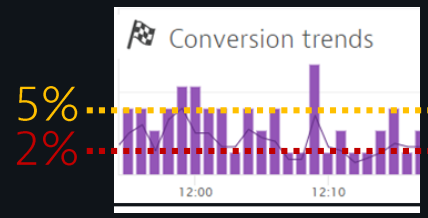
Grader

```
{
  "spec_version" : "1.0",
  "indicators": [
    {
      "id": "go_memstats_alloc_bytes",
      "source": "Prometheus",
      "query": "avg(go_gc_duration_seconds{instance='localhost:909
      "output" : "total",
      "grading": {
        "type": "Threshold",
        "thresholds": {
          "upperSevere": 21802344
        },
        "metricScore": 20
      }
    },
    {
      "id": "ConversationRate",
      "source": "Dynatrace",
      "query": {
        "timeseriesId": "com.dynatrace.builtin:app.conversionrate"
        "aggregation": "avg",
        "tags": [
          "SockShop-Blue"
        ]
      },
      "output": "series",
      "grading": {
        "type": "Threshold",
        "thresholds": {
          "lowerSevere": 2,
          "lowerWarning": 5
        },
        "metricScore": 20
      }
    },
  ],
  "objectives": {
    "pass": 30,
    "warning": 15
  }
}
```



> 2GB: 0 Points

< 2GB: 20 Points

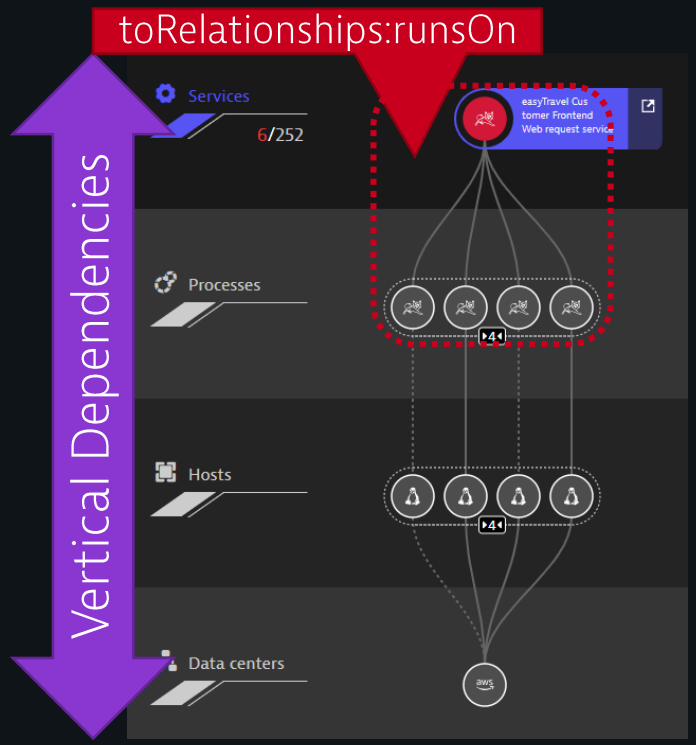


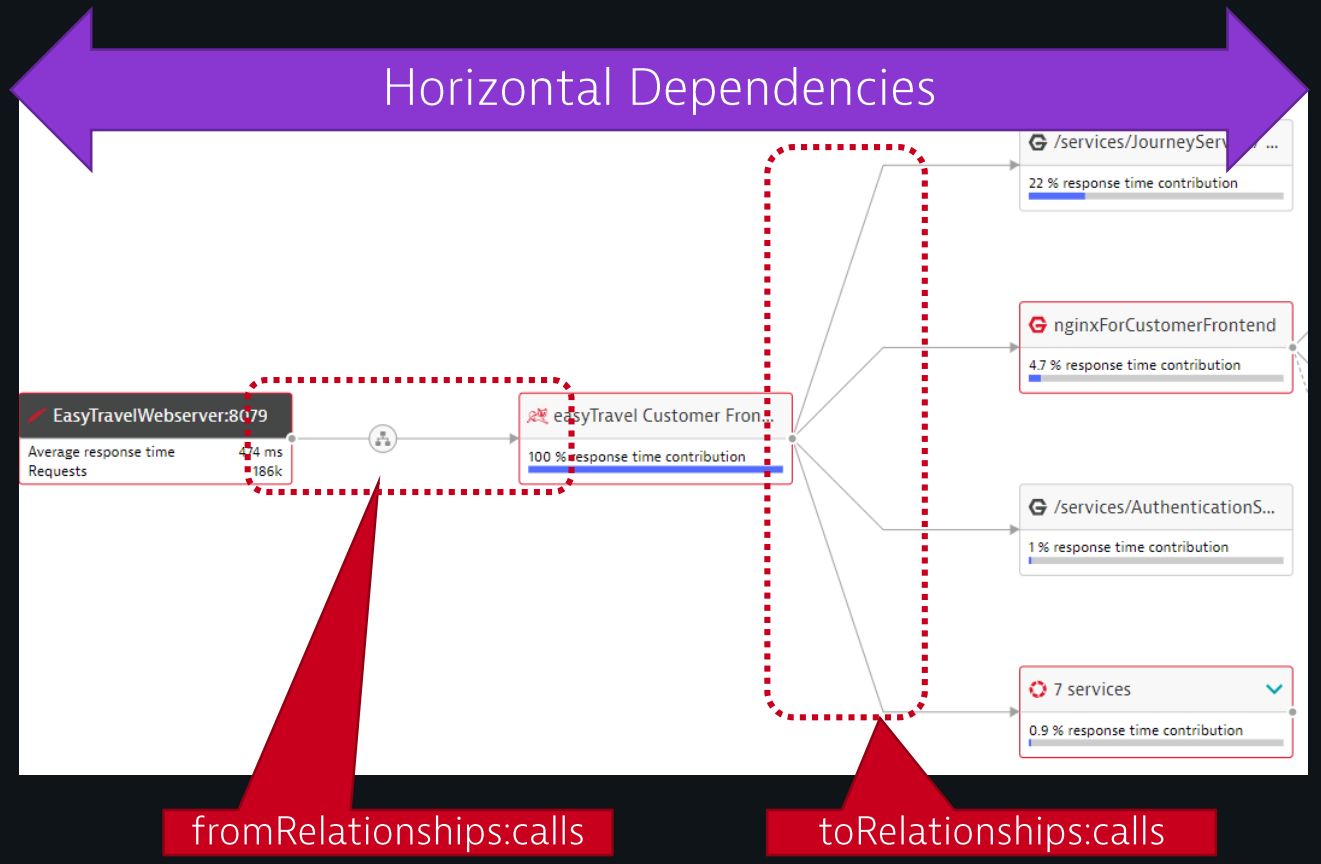
> 5%: 20 Points

< 5%: 10 Points

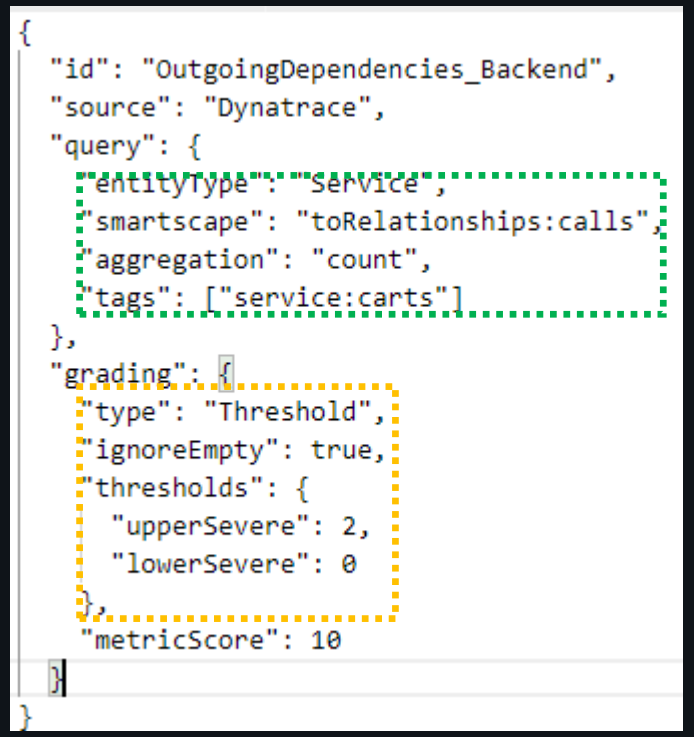
< 2%: 0 Points

Pitometer: Dynatrace Smartscape Data Source





Pitometer: Dynatrace Smartscape Data Source Example



```
{
  "id": "OutgoingDependencies_Backend",
  "source": "Dynatrace",
  "query": {
    "entityType": "SERVICE",
    "smartscape": "toRelationships:calls",
    "aggregation": "count",
    "tags": ["service:carts"]
  },
  "grading": {
    "type": "Threshold",
    "ignoreEmpty": true,
    "thresholds": {
      "upperSevere": 2,
      "lowerSevere": 0
    },
    "metricScore": 10
  }
}
```

Pitometer: Run Standalone - https://github.com/keptn/pitometer

Init

Source

Source

Grader

Run

```
const Pitometer = require('@pitometer/pitometer').Pitometer;
const DynatraceSource = require('@pitometer/source-dynatrace').Source;
const PrometheusSource = require('@pitometer/source-prometheus').Source;
const ThresholdGrader = require('@pitometer/grader-threshold').Grader;

const pitometer = new Pitometer();

// Register a Prometheus source that will be used if the source ID in your
// perfspec matches 'Prometheus'
pitometer.addSource('Prometheus', new PrometheusSource({
  queryUrl: '<PROMETHEUS_PROMQL_ENDPOINT>',
}));

// Register a source that will be used if the source ID in your perfspec matches
// 'Dynatrace'
pitometer.addSource('Dynatrace', new DynatraceSource({
  baseUrl: '<DYNATRACE_ENVIRONMENT_URL>',
  apiToken: '<DYNATRACE_API_TOKEN>',
  // Optional: A logger to be used for debugging API requests
  // log: console.log,
}));

// Register a grader for thresholds that will be used if the grader type
// matches 'Threshold'
pitometer.addGrader('Threshold', new ThresholdGrader());

// Load a perfspec - see the samples directory
const perfspec = require('./samples/monspec-sample.json');

// Run the perfspec, apssing in an optional context parameter 'prod'
// and log the result out to the console
pitometer.run(perfspec, 'prod')
  .then((results) => console.log(JSON.stringify(results)))
  .catch((err) => console.error(err));
```

Result


```
{
  "totalScore": 10,
  "objectives": {
    "pass": 30,
    "warning": 15
  },
  "indicatorResults": [
    {
      "id": "go_memstats_alloc_bytes",
      "score": 0,
      "violations": [
        {
          "value": 39803321,
          "key": "localhost:9090",
          "breach": "upper_critical",
          "comparison": "fixed",
          "threshold": 21802344
        }
      ]
    },
    {
      "id": "ConversationRate",
      "score": 10,
      "violations": [
        {
          "value": 3.9,
          "key": "SockShop-Blue",
          "breach": "lower_warning",
          "comparison": "fixed",
          "threshold": 5
        }
      ]
    }
  ]
}
```

Pitometer in keptn

Dev

1: push

2: deploy (shadow)

3: test (performance)

4: evaluate (scalability KPIs)

5: promote

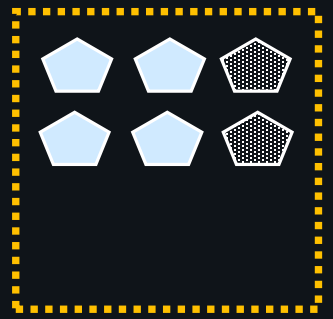
6: deploy (blue/green)

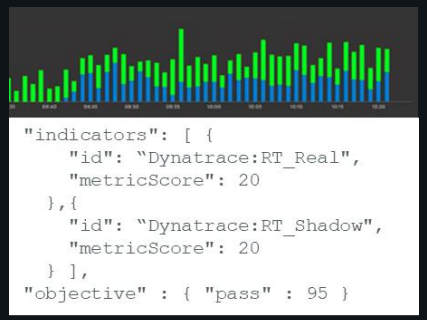
7: evaluate (business KPIs)

8: operate (NoOps)

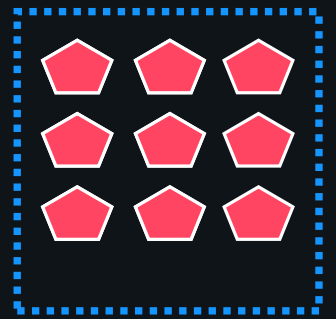
keptn Autonomous Cloud Control Plane

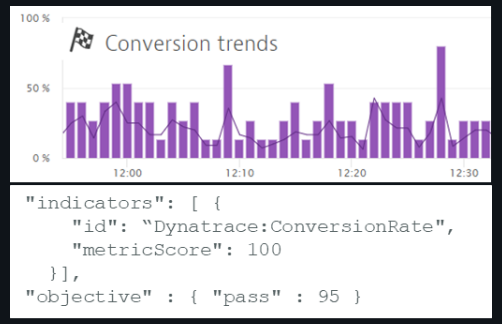
stage





prod





keptn.sh - OpenSource framework for unbreakable pipeline and more ...

Enterprise-grade framework for shipping and running cloud-native applications

Deployable on any Kubernetes cluster, keptn converts any Kubernetes cluster into a self-healing, autonomous cloud fabric.

keptn

CORE CAPABILITIES

- Automated multi-stage unbreakable delivery pipelines
- Self-healing blue / green deployments
- Event-driven runbook automation

DESIGN PRINCIPALS

- GitOps-based collaboration
- Operator patterns for all logic components
- Monitoring-and-operations-as-code

- Built-on and for Kubernetes
- Event-driven and serverless
- Pluggable tooling

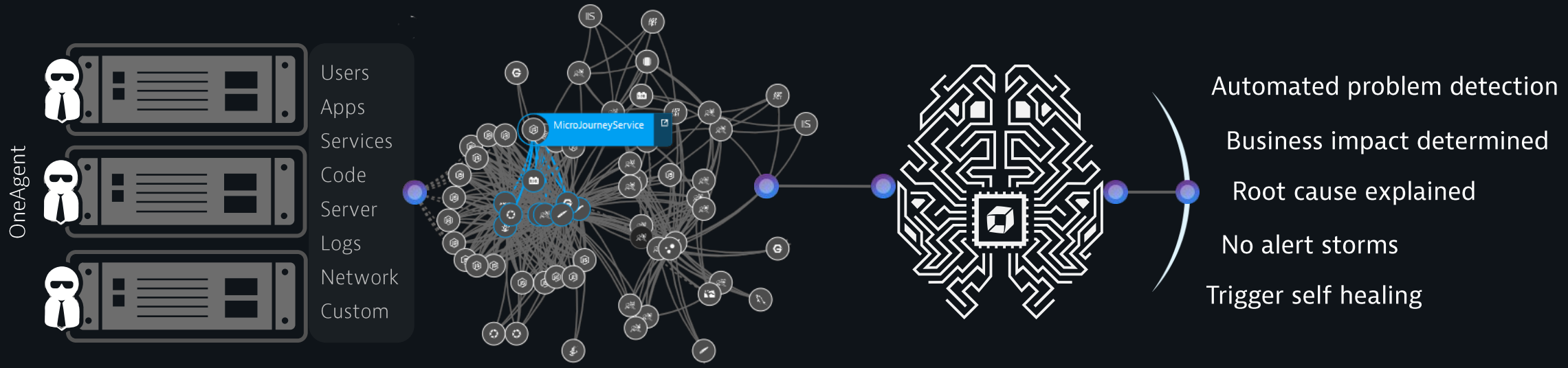
Resources

- Keptn & Pitometer
  - www.keptn.sh
  - github.com/keptn
  - github.com/keptn/pitometer
- Performance, Resiliency & Availablity Content
  - Adrian Hornsby (AWS): https://speakerdeck.com/adhorn/resiliency-and-availability-design-patterns-3742b5ba-e013-4f50-8512-00a65775f478
  - Acacio Cruz (Google): https://www.spreaker.com/user/pureperformance/066-load-shedding-sre-at-google-with-aca
  - Thomas Steinmaurer (Dynatrace): https://www.neotys.com/performance-advisory-council/thomas_steinmaurer

Demo Time

Dynatrace AI-powered better data makes automation & unbreakable CD possible

High fidelity data | Mapped end-to-end | Deterministic AI | Answers + Action



Completely automated

Sample Dynatrace global DevOps customers

Speaking Sessions (recap)

- The machines are coming, but you have nothing to fear
  - Day/Time: Tuesday, May 7, 9:30 a.m.-10:15 a.m.
  - Speaker: Dave Anderson
- Building autonomous operations for Kubernetes with keptn
  - Day/Time: Tuesday, May 7, 2:30 p.m.-3:15 p.m.
  - Speaker: Alois Reitbauer
- Shift-left site reliability engineering for self-healing applications
  - Day/Time: Wednesday, May 8, 10:30 a.m.-11:15 a.m.
  - Speaker: Jürgen Etzlstorfer
- Unifying OpenShift cluster, container and application monitoring
  - Day/Time: Wednesday, May 8, 3:30 p.m.-3:50 p.m.
  - Speaker: Asad Ali
- Unbreakable DevOps on Red Hat OpenShift
  - Day/Time: Thursday, May 9, 1:00 p.m.-1:45 p.m.
  - Speaker: Peter Hack
- Full speaking session information available HERE

Stay engaged with us!

- Download a free copy of the Gartner MQ http://bit.ly/2FjBzcC
- Take our survey https://dynatrace.ai/acsurvey
- Catch our PurePerformance Podcast Series https://www.spreaker.com/user/pureperformance
- Check out our Performance Clinics https://bit.ly/2pzXXIK
- Try our free trial! https://www.dynatrace.com/trial/

dynatrace.com